**Mixing Desk Development Evaluation Report**

Michael Johnston

(1800329)

Table of Contents

[Introduction 3](#_Toc39663428)

[Functional and Non-Functional Requirements 3](#_Toc39663429)

[Functional 4](#_Toc39663430)

[Non-Functional 8](#_Toc39663431)

[Strengths and Weaknesses in the Development Stage 9](#_Toc39663432)

[Strengths 9](#_Toc39663433)

[Weaknesses 10](#_Toc39663434)

[Future Development Recommendations 11](#_Toc39663435)

[App 11](#_Toc39663436)

[Mixing Calculator section 11](#_Toc39663437)

[Database 12](#_Toc39663438)

[Recipes 12](#_Toc39663439)

[Cosmetic 13](#_Toc39663440)

[Changes to the Solution Design and Project Plan During Implementation 14](#_Toc39663441)

[Knowledge and Skills Gained During Development and Recommendations on Improving the Development Process 15](#_Toc39663442)

[How the Project Process Could Have Been Improved 16](#_Toc39663443)

[Conclusion 16](#_Toc39663444)

# Introduction

This report is intended to evaluate the development of the Mixing Desk app that was produced by this developer and will scrutinise the following aspects of the development phase:

* How well the Functional and Non-Functional Requirements were met
* Strengths and weaknesses in the development phase
* Future development recommendations
* A summary of modifications that were made during development
* Identification of the knowledge and skills gained

# Functional and Non-Functional Requirements

The functional and non-functional requirements for the Mixing Desk app devised during the inception phase of this project were as follows:

**Functional**

* The user must be able to calculate the weights of each necessary ingredient in a recipe to make a complete e-liquid.
* The user must be able to browse a list of stock recipes (which will come with the app) and filter and display them for a given range of post-dates (i.e. past month, past year, past week etc).
* The user must be able to sort the recipe list by recipe rating
* The user must be able to maintain a list of the flavours they have (stock flavours will come with the app) and add new ones.
* The user must be able to save their own recipes and add it to the recipe pool if they wish to do so.
* The app must be easy to navigate.

**Non-Functional**

* The app must be completed and delivered along with an evaluation by the 30th of April 2020.
* The app must store data in a local database.
* The app must run on Windows 10.

We will now analyse each of these requirements and discuss why they were or were not met.

## Functional

**The user must be able to calculate the weights of each necessary ingredient in a recipe to make a complete e-liquid**

This requirement was fulfilled. The finished app is capable of calculating e-liquid mixes for any recipe given that it the recipe has 7 flavours or less. Figure 1 below shows the app calculating a recipe brought in from the recipes section of the app.

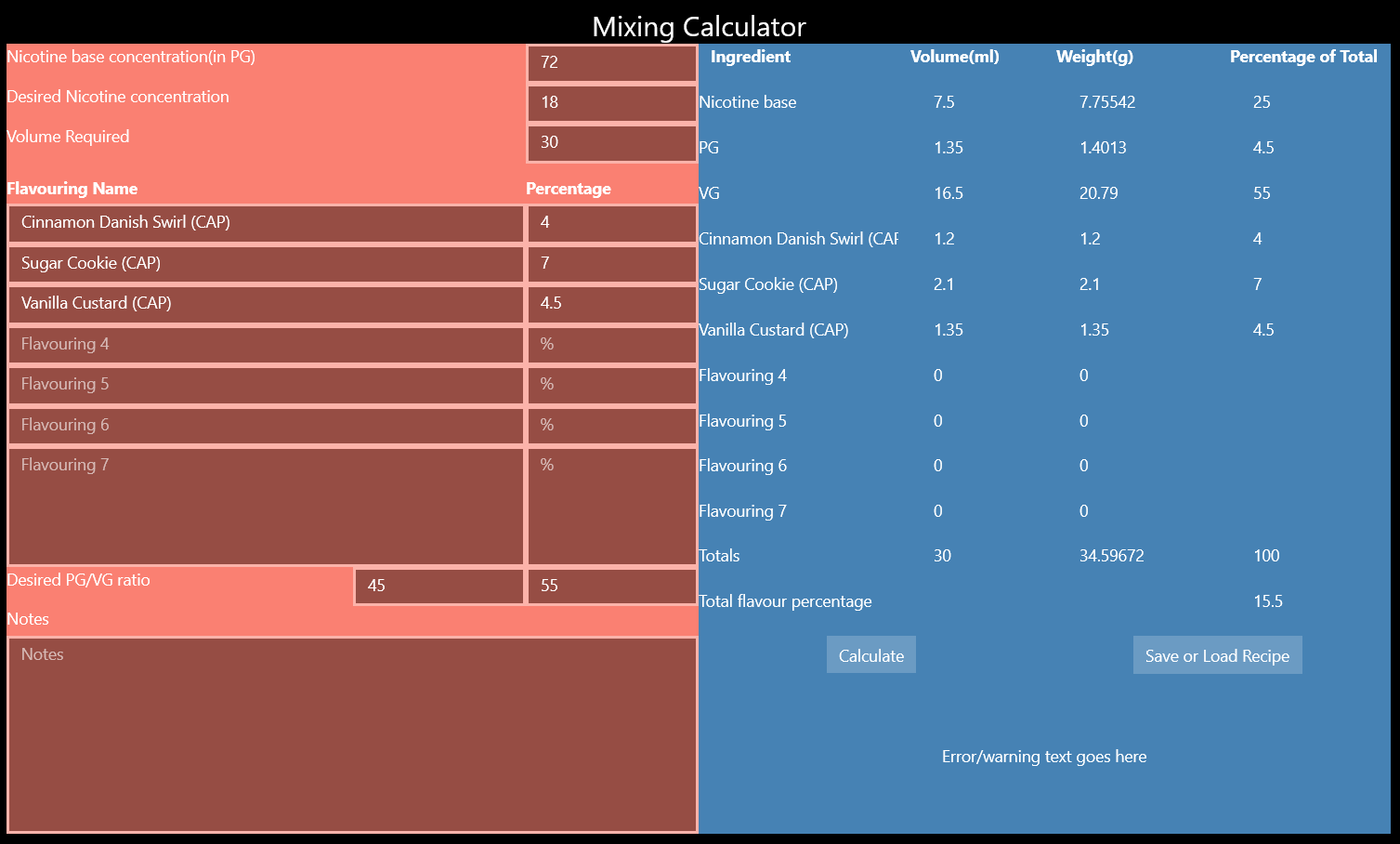


Figure : Mixing Calculator page calculating recipe with less than seven flavourings.

The output on this page could be better presented. this is particularly apparent when calculating a recipe with less than seven flavourings as the redundant output fields are missing entirely. Though this is purely a cosmetic issue, the calculation is correct.

In the development of the app this section was developed to provide the right numbers and no more. Development time only allowed for the error catching in this section to be "good enough" for basic input. There is currently an input output mismatch that will occur if a flavouring is added to one of the fields if the field used is not immediately after the last filled field. An example of this is shown in figure 2.

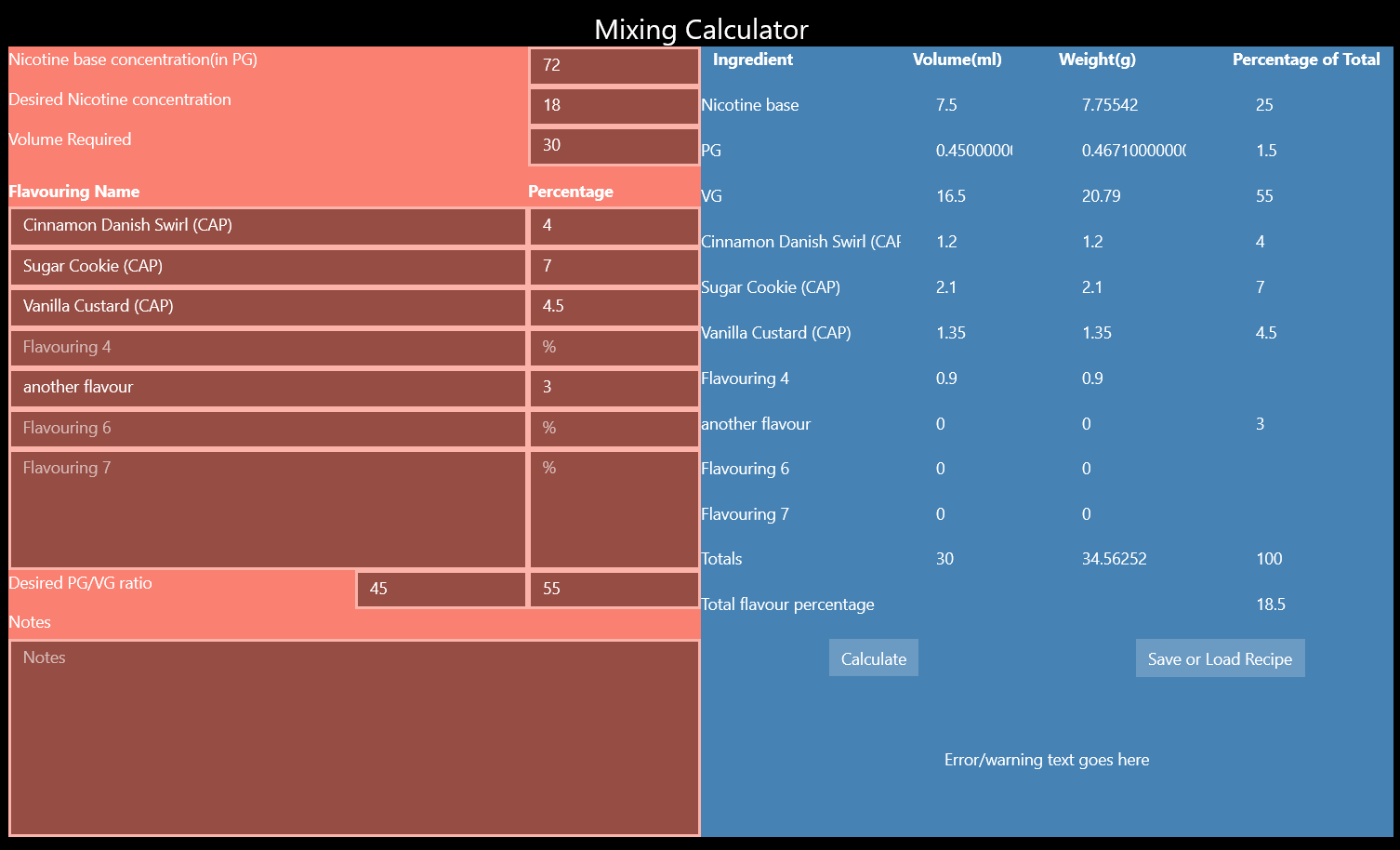


Figure : Mixing Calculator mismatched output.

The additional 0.9 in the output is the correct number (as shown in figure 3) it is just in the wrong place. This could be very misleading to users and would need immediate fixing in further development. This was likely caused by a lack of attention when developing the output functions.

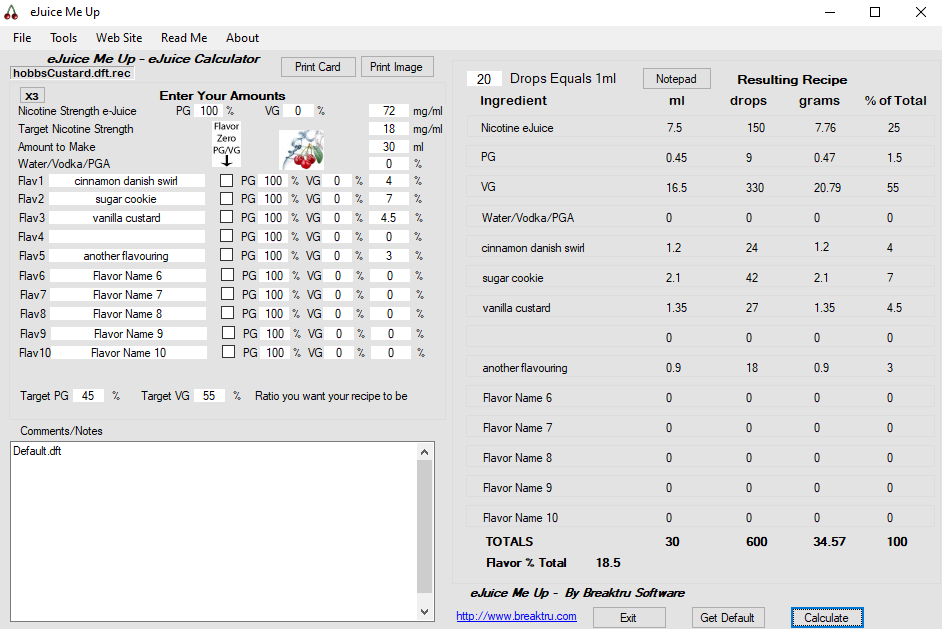


Figure : eJuice Me Up showing the correct numbers.

**The user must be able to browse a list of stock recipes (which will come with the app) and filter and display them for a given range of post-dates (i.e. past month, past year, past week etc).**

This requirement has been fulfilled. Through the usage of the integrated Sqlite database and the three recipe pages users can browse recipes from the past year, month or week. They can also sort these lists by date by clicking the date column header on the lists to sort by descending date.

**The user must be able to sort the recipe list by recipe rating.**

This has also been fulfilled. The user can click the rating column header on each recipe list to sort the recipes by descending rating.

**The user must be able to maintain a list of the flavours they have (stock flavours will come with the app) and add new ones.**

This requirement has been partially fulfilled. The user can add flavourings the personal stash section though anything added will not be persistent across restarts of the app. To make this persistent the flavourings in the stash would perhaps have to be stored in some kind of app setting or as a saved file. This developer was only beginning to research those topics when testing had to start and as such did not have time to implement a persistent flavour stash. Though the current behaviour of this section is more than sufficient as a proof of concept.

**The user must be able to save their own recipes and add it to the recipe pool if they wish to do so.**

This requirement has also been partially fulfilled. Users can save or load recipes in .txt format from the Mixing Calculator page by using a standard file picker. There is currently no facility allowing the user to send their custom recipe to the recipe section of the app. While this facility would have been relatively easy to develop, the choice was made to implement the saving and loading of recipes locally as this was seen as a more fundamental and necessary feature. This is because the save files are persistent on the user's computer whilst any additions to the recipe section would not be persistent. This is yet another demonstration that one of the major improvements to this app that could be made is to make all database aspects persistent, perhaps as a database located on the internet, contacted by the app.

**The app must be easy to navigate.**

This requirement is considered to be fulfilled. Due to the navigation bar on the left of the app with appropriate names of each section of the app listed the user can navigate easily to each page of the app.

## Non-Functional

**The app must be completed and delivered along with an evaluation by the 30th of April 2020.**

This requirement is fulfilled. The finished prototype application was delivered on the 15th of April 2020 and this evaluation of the app's development will be delivered before the new deadline for the evaluation of the 7th of May 2020.

**The app must store data in a local database.**

This requirement is fulfilled. The app stores data in an integrated Sqlite database.

**The app must run on Windows 10.**

This requirement is fulfilled. The app is built using Universal Windows Platform and C# both of which are highly compatible with windows 10. The app was also tested on Windows 10.

# Strengths and Weaknesses in the Development Stage

We will now discuss some strengths and weaknesses that have been identified during the development of the prototype application.

## Strengths

**Coding of the Business Model**

The coding of the business model was generally quick and efficient due in part to a well thought out business model in the design stage.

**User Documentation**

User documentation was well handled with a traditional user manual which was exhaustive in elucidating to the user the features and navigation of the app.

**Error Handling**

Error handling was competently achieved in the app where it was necessary with try-catch blocks and input validation in a few different forms was also present. Some custom exceptions would have enhanced this.

**Test plan**

The test plan was described in the development report and evidence produced for both black and white box testing of the app.

**Using unfamiliar constructs and libraries**

Many unfamiliar constructs and libraries were utilized in the development of the app and were generally easily incorporated into the app. Some examples include the integration of Sqlite, context menus and saving and loading files.

## Weaknesses

**Evaluating test runs**

Unit testing test runs were not evaluated and no recommendations for future testing were given.

**Coding of the user interface domain**

This is included as a weakness regarding the aesthetics of the app's interface. Very little effort was given to improving the aesthetics of the app due in large part to time constraints and the need to develop the functionality of the app.

**Internal Documentation**

Internal documentation was generally good regarding comments and doc-strings for not so easily discerned functions and sections of code. This is included as a weakness due to some inconsistencies in event-handler method names beginning with or without capital letters.

# Future Development Recommendations

In both the development and evaluation stage of this project many recommendations can now be made as to how to develop the application further. These recommendations will now be listed (with reasoning the behind them) though this list is not exhaustive.

## App

**Fully deploy the app to a store**

This would allow the user to use the app without having to use visual studio.

**Personal user accounts**

In order to store a user's personal flavourings, the app would have to allow the user to sign into an account. This would also facilitate many other features such as users having favourite recipes and users posting recipes.

## Mixing Calculator section

**Fix output in mixing calculator section**

As discussed previously the output in this section of the app is not as presentable as it could be, and certain inputs lead to misleading outputs.

**Allow notes to be saved when recipes are saved.**

Notes are an important tool when people are developing recipes and users must be able to see what their previous thoughts on the recipe were.

**Bind the PG/VG input text boxes to each other to improve ease of use**

Currently these two input textboxes must add up to 100 or an error is thrown. Changing one when the other is changed will allow the app to be used faster and more effectively.

**Allow nicotine base to be in VG as well as PG**

Nicotine base is often sold in a vegetable glycerine solution instead of propylene glycol. This would increase the usefulness of the app to users who have their nicotine base in this form.

## Database

**Make the database persistent and located on the internet**

This is the single greatest improvement to the app that could be made. The database would be persistent across restarts of the app and this would improve both the recipes and personal stash sections.

**Allow for the sharing of a custom recipe to the database**

This would allow users to share recipes with each other and would fulfil the relevant functional requirement.

## Recipes

**Allow list sorting by date and rating to sort by ascending on second click of the column headers**

This would enhance the app's functionality to a level that the user expects of it. The user could easily find the oldest or lowest rated recipe.

**Allow list sorting by name of recipe**

This would make finding recipes easier.

**Create a new Recipes page for recipes from all time**

With the current range of recipe pages a user cannot view recipes created more than a year ago.

## Cosmetic

**Implement a colour scheme for the interface**

The current colour scheme is bland and uninteresting and unhelpful in allowing the eye to differentiate the different components of the interface.

**Improve the spacing and layout of the interface**

This would further improve the look and feel of the interface enhancing the user experience.

# Changes to the Solution Design and Project Plan During Implementation

In the development of this application some elements of the design had to be changed though in general the initial design was efficacious. What follows is a summary of the changes to the project plan and the solution design and implementation.

The development phase was originally intended to begin on the 13th of February 2020 though was delayed to the 27th of February to extend time on the solution planning phase.

When first beginning development a test project was made to test and learn how to integrate a Sqlite database with a UWP app. This was expected to go smoothly though the tutorial used for this did not work. This unforeseen event was corrected in a few days after following the tutorial while consulting another instruction set.

Very little was changed regarding the design of the business model and its implementation with most of the designed methods of each class integrated into the properties of each class. The ingredient class was changed in that its flavouring member variable was changed from a flavouring type to a simple string. This removes the tight coupling it had with the flavouring class.

Classes in general also did not need the idnumber property that was expected.

The user interface of the finished app is almost identical to the design. Though it was discovered during development that list view elements do not include any header features for columns etc. This was compensated for by adding interactive text blocks above the list view elements, some of which behave in the same way as column headers for sorting.

The Mixing Calculator page was changed slightly to allow for a button for saving or loading recipes and to show the volume in millilitres of proportions of mixing ingredients.

# Knowledge and Skills Gained During Development and Recommendations on Improving the Development Process

In the process of developing the Mixing Desk app new knowledge and skills were gained. We will now discuss some of them and outline how and why they were gained.

**A Better understanding of UWP**

This developer’s knowledge of Universal Windows Platform was widened significantly through the development of this application. This happened through reading more of Microsoft's documentation and incorporating what was learned into the app. This had to be done to include previously promised functionality for the app.

**Integrating Sqlite into a UWP app**

Prior to developing this app this developer was unaware of how to integrate a database into a UWP app. This was achieved by following tutorials and reading Microsoft's UWP documentation. This had to be achieved to fulfil the functional requirements of the app.

**Testing documentation**

Prior to developing this app this developer had never produced such thorough and detailed testing documentation. In the process we gained better knowledge of the testing cycle in development and the testing process internal details. This had to be achieved to provide assurance that the software worked as intended.

**Data binding and data templates**

Prior to developing this app this developer was unaware of how to employ data binding in UWP to display data. We were also unaware of data templates and how to construct them to display data. This was achieved by following tutorials and consulting instructional videos on the topic. This had to be achieved in order to display the data the app is working with from the database in an orderly manner.

## How the Project Process Could Have Been Improved

We have at many points in this evaluation mentioned that many things were not achieved due to lack of development time. The entire project process could have been improved by not delaying the start date of development. This could have resulted in a complete fulfilment of the functional requirements and some cosmetic improvements may have been achieved.

This developer is of the opinion that the most significant change that could be made to improve the development process is to have been aware and practiced in the unfamiliar elements of the app before the development process began (for example being practised in integrating Sqlite with UWP prior to starting development). This would have made development faster and thus more could have been done.

# Conclusion

In summary, this evaluation has shown that the project to produce the Mixing Desk app went very well and the development phase was largely successful. Almost all functional and non-functional requirements were accomplished, we have evaluated the strengths and weaknesses of the development phase, provided some useful recommendations for future development and identified the knowledge and skills that were gained from the development phase. These recommendations and insights will be taken forward into future iterations of this software.